Les protocoles réseaux de bavardage

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 juin 2009

http://www.bortzmeyer.org/gossip-protocol.html

Il existe une classe de protocoles réseaux nommés protocoles de **bavardage** ("*gossip protocol*") qui résoud élegamment un problème classique dans les réseaux : informer tout le monde sans utiliser de "*mass media*" centralisé. Ils utilisent pour cela un principe simple : tout pair du réseau transmet l'information aux pairs qu'il connait (un sous-ensemble du total) et ceux-ci, à leur tour, transmettent aux pairs qu'ils connaissent, et ainsi de suite (on appelle parfois cette procédure l'**indondation** - "*flooding*"). Au bout d'un moment, la totalité du réseau est au courant, sans qu'on aie eu besoin d'un système centralisé.

L'Internet utilise un certain nombre de protocoles de bavardage. C'est ainsi que le routage global, avec le protocole BGP (RFC 4271 [1]), dépend d'un protocole de bavardage. Aucun routeur BGP ne connait tous les autres routeurs BGP de la DFZ, l'Internet n'a pas de centre, pas de chaîne de télévision unique qui informerait tous les routeurs de l'arrivée d'un nouveau réseau. Pourtant, de fil en aiguille, l'information sur un nouveau préfixe d'adresses IP touche tout l'Internet en quelques minutes.

Avant BGP, les protocoles de bavardage ont été surtout popularisés par les "*News*" (RFC 5537 et RFC 3977), où un article était diffusé à chaque pair (par opposition au Web qui est d'avantage centralisé : si le serveur de `www.playboy.fr` tombe en panne ou bien est victime d'une attaque, l'information (?) sur le site n'est plus accessible du tout). Au contraire, avec BGP ou les "*News*", il faut détruire beaucoup de machines pour réellement supprimer certaines informations ! (Avec les protocoles de bavardage, l'information cesse d'être accessible à tous dès que le réseau cesse d'être connecté, lorsqu'il est **partitionné** en deux - ou plus - morceaux.)

Donc, les principales forces des protocoles de bavardage sont la résistance aux pannes et le fait qu'aucun nœud ne joue de rôle essentiel, y compris pour l'injection initiale (alors que le terme d'« inondation » n'implique pas d'égalité, l'inondation peut aussi servir lorsqu'un des nœuds est le seul à pouvoir initier une information).

1. Pour voir le RFC de numéro NNN, `http://www.ietf.org/rfc/rfcNNN.txt`, par exemple `http://www.ietf.org/rfc/rfc4271.txt`

Pour fonctionner correctement, les protocoles de bavardage ont besoin que chaque pair maintienne un **historique** : quels messages a t-il reçus, a qui les a t-il envoyés, etc. Autrement, on perdrait du temps et des ressources à retransmettre un message déjà connu et on risquerait même de faire des boucles sans fin. Dans les "*News*", chaque serveur retient les messages reçus (identifiés par leur `Message-ID:`, et « oubliés » au bout d'une durée réglable, pour éviter de remplir le disque dur). BGP a en outre d'autres mesures pour limiter l'inondation comme le fait de ne transmettre comme toutes que celles qu'il utilise lui-même, éliminant ainsi d'une propagation ultérieure les « mauvaises » routes.

Une propriété intéressante des architectures fondées sur le bavardage est qu'il n'est pas nécessaire que la transmission des messages entre les pairs utilise toujours le même protocole. Par exemple, dans les "*News*", le protocole NNTP (RFC 3977) est le plus courant mais pas le seul (pendant longtemps, UUCP était plus utilisé).

Pour illustrer le fonctionnement d'un protocole de bavardage, voici une petite implémentation en Python, avec la bibliothèque standard socket <`http://docs.python.org/library/socket.html`> : (en ligne sur `http://www.bortzmeyer.org/files/gossiper.py`). Les fils sont utilisés pour éviter d'être bloqué, chaque pair pouvant évidemment être en panne, injoignable, etc. Dans l'implémentation actuelle, on réessaie un nombre limité de fois si un pair n'est pas joignable. Il ne faut évidemment pas comparer cette petite implémentation aux vraies qui tournent sur le vrai Internet, notamment parce qu'elle manque sérieusement de robustesse, par exemple lors de la lecture des messages des autres pairs.

Dans cet exemple simple, le protocole entre deux pairs est fixé (comme avec BGP, mais contrairement aux "*News*"). Chaque pair a un numéro unique, l'**ID**, fixé à la main au démarrage (si on n'a pas de serveur central pour attribuer des numéros uniques, le mieux est de tirer ces ID au hasard dans un espace très grand, limitant ainsi les risques de collision, ou bien de prendre le résumé cryptographique d'une phrase, unique pour chaque pair.) Le « serveur » (le pair qui répond aux demandes de connexion) envoie son ID (suivi d'une virgule). Le « client » (le pair qui lance la connexion) envoie son ID, suivi d'une virgule et suivi du message (sur une seule ligne). Si l'ID est l'**identificateur** du pair, pour s'y connecter effectivement sur l'Internet, il faut un **localisateur**, un tuple (adresse IP, port). Dans cette simple implémentation, il n'y a qu'un de ces tuples par pair mais ce n'est pas imposé par le protocole (de toute façon, les pairs sont identifiés par leur ID, pas par leur adresse IP, donc une connexion du pair N peut venir d'une autre adresse IP que celle connue pour N, si la machine cliente a plusieurs adresses).

Au démarrage, on indique donc au programme (en ligne sur `http://www.bortzmeyer.org/files/gossiper.py`) le port sur lequel il doit écouter (30480 par défaut) et les identificateurs et localisateurs de ses pairs. À noter que la relation entre pair n'est pas symétrique : 1 peut savoir comment joindre 2 sans que le contraire soit vrai. Le nombre de pairs est quelconque mais il faut juste que l'ensemble des pairs soit connecté sinon certains messages n'arriveront pas partout.

Les pairs se souviennent des messagers qu'ils ont vu (dans l'historique global) et de ceux qu'ils ont envoyés à chaque pair (dans un historique spécifique au pair).

Le mode d'emploi est simple. Supposons qu'on veuille lancer le pair d'ID 42, écoutant sur le port 22000, et pouvant parler aux pairs d'ID 261 (`2001:DB8:1::bad:dcaf`, port par défaut) et d'ID 4231 (`2001:DB8:2::dead:beef`, port 443), on tapera :

```
% python gossiper.py -i 42 -p 22000 \
    261,\[2001:DB8:1::bad:dcaf\] 4231,\[2001:DB8:2::dead:beef\]:443
```

La syntaxe (adresse IP, port) utilisée est celle du RFC 3986, section 3.2.2. Les barres obliques inverses devant les crochets sont imposés par certains shells Unix.

Le programme affichera alors les communications avec ses pairs. Pour amorcer la pompe et injecter des messages facilement <http://www.bortzmeyer.org/commande-rappelable-pour-tests.html>, on peut utiliser netcat (ici, l'injecteur original a indiqué un ID de 0) :

```
% echo 0,"Les sanglots longs des violons de l'automne" | nc 127.0.0.1 22000
```

Avec des messages inspirés de Radio Londres, on obtiendra des dialogues comme suit (un délai artificiel a été ajouté pour mieux simuler la durée de propagation dans un réseau). Ici, le pair d'ID 5 :

```
2009-06-05 17:30:13 - ::ffff:127.0.0.1 - 30 bytes (NEW message received from peer 0: "Jean a de longues moustach
2009-06-05 17:30:29 -  - 0 bytes (Sender task Thread-1 received "Jean a de longues moustaches" from 0, connectin
```

Et ici le 2 :

```
2009-06-05 17:31:30 - ::ffff:192.134.4.69 - 30 bytes (NEW message received from peer 5: "Jean a de longues moust
2009-06-05 17:32:01 -  - 0 bytes (Sender task Thread-3 received "Jean a de longues moustaches" from 5, connectin
```

Et ici, un pair, le 3, reçoit un message deux fois, de 6, puis de 2. il ignore le second. Cette précaution est ce qui permet aux protocoles de bavardage de ne pas boucler éternellement.

```
2009-06-06 22:56:35 2a01:e35:8bd9:8bb0:21c:23ff:fe00:6b7f - 30 bytes - NEW message received from peer 6: "Jean a
2009-06-06 22:56:49 2a01:e35:8bd9:8bb0:21c:23ff:fe00:6b7f - 30 bytes - Ignoring known message from peer 2: "Jean
...
2009-06-06 22:58:08 2a01:e35:8bd9:8bb0:21c:23ff:fe00:6b7f - 35 bytes - NEW message received from peer 6: "Les la
2009-06-06 22:58:28 2a01:e35:8bd9:8bb0:21c:23ff:fe00:6b7f - 35 bytes - Ignoring known message from peer 2: "Les
```

Pour approfondir le concept, on peut lire le bon article du Wikipédia anglophone.